PCT

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(54) Title: QUASI RADIX-16 PROCESSOR AND METHOD

(57) Abstract

A Quasi Radix-16 Butterfly comprises a radix-4 butterfly processor and on-board memory with external memory addressing changes from a conventional radix-4 butterfly processor. On-chip cache memory is included to store data outputs of the radix-4 butterfly processor for application as data inputs to the radix-4 butterfly processor in a second series of butterfly operations to implement high-speed processing that is maximally execution-bound.

-DESCRIPTION

QUASI RADIX-16 PROCESSOR AND METHOD

Background of the Invention

5   1.   Field of the Invention

The present invention relates to a LSI butterfly

processor and method for high performance Fast Fourier

Transform (FFT) processors.

10   2.   Description of Related Art

With the present state of the art in VLSI

technology, the speed of VLSI computing structures is

typically limited by the data bandwidth of a VLSI array

processor and not by silicon circuits. The data bandwidth is

15   directly related to the number of input and output (I/O)

pins. The level of integration has reached a point where,

due to limited I/O bandwidth, it is not possible to achieve

100% execution hardware utilization without increasing the

number of I/O pins significantly. In such a situation, the

20   design of VLSI processors should aim for minimization of the

undesirable effects of limited I/O bandwidth while preserving

the tremendous advantages of large computation arrays.

Early integrated FFT processors were based on a

radix-2 "butterfly" as illustrated in Figure 1a, to keep the

25   amount of hardware required on a single chip to a minimum. A

"butterfly" is a set of arithmetic operations commonly used

in digital signal processing. Recently, due to advancing

technology, several radix-4 butterfly-based processors have been disclosed. These processors are a logical evolutionary advance from radix-2 processors that takes advantage of a higher level of integration and thus more silicon area. Each radix-4 butterfly requires four data inputs, three 'twiddle' inputs (i.e. angular velocity information), and four data outputs, as shown in Figure 1b. A typical multi-cycle radix-4 butterfly processor includes one complex data input port, one complex twiddle port and one complex data output port, where each port can be 32 to 48 bits wide in such fixed point processors. Thus, for a radix-4 butterfly processor there is already a large number of I/O pins. Also, it is difficult to conceive of a radix-8 butterfly processor because there is a tremendous increase in the amount of hardware required for a radix-8 butterfly processor as compared with a radix-4 butterfly processor. The term "butterfly" processor is used herein to conveniently designate the circuit module, described in detail later herein, which includes a plurality of fan-in inputs and a plurality of fan-out outputs in the schematic representation.

It is desirable to compute Fast Fourier transforms (FFTs) using as high a radix as possible to alleviate the I/O bandwidth problem in VLSI FFT processors. The use of higher radices is even more desirable for FFT processors designed to handle exceptionally large FFT sizes e.g., 16 million data-point FFT. The I/O bandwidth problem in high performance FFT processors is illustrated in the timing

diagrams of Figures 2a and 2b. Let Tio be the I/O cycle
time and Teu be the execution unit pipeline cycle time. For
a dual radix-2 butterfly, four I/O cycles are performed to
fetch the data (A1, B1, A2, B2), as shown in Figure 2a. The
5   datapath then performs two butterfly operations (1,2) on
fetched data in two cycles. If Tio is equal to Teu then the
datapath cannot be kept busy continuously and is I/O bound as
indicated by the blank cycles in the datapath timing waveform
of Figure 2a. As indicated in Figure 2a, to make the
10 datapath execution bound, $4 \times Tio = 2 \times Teu$ i.e., the I/O cycle
time should have to be one-half the execution pipeline cycle
time. Figure 2b illustrates the same situation for a radix-4
butterfly. Here again four I/O cycles are required to fetch
the data for a single radix-4 butterfly. If it is assumed
15 that the datapath can execute a radix-4 butterfly every two
cycles, then once again Tio would have to be (1/2) Teu to
make the datapath execution bound. This then is the I/O
bandwidth problem in high performance pipelined FFT
processors with a limited number of I/O ports, specifically
20 three complex ports (namely, an input port, a twiddle port
and an output port as indicated in Figure 2).

                        Summary of the Invention

        In accordance with the illustrated embodiment of
25 the present invention, radix-16 butterfly hardware is
implemented using a multi-cycle radix-4 butterfly-based
circuit module that uses relatively fewer I/O pins and yet

relieves the I/O bandwidth problem to achieve significantly higher throughput compared with a conventional radix-4 butterfly implementation. This invention, henceforth referred to as the Quasi Radix-16 (QR16) butterfly processor, includes a radix-4 butterfly processor and on-board memory with some minor external memory addressing changes from a conventional radix-4 butterfly processor. The QR16 performs a quasi radix-16 butterfly in two columns of four radix-4 butterfly operations as illustrated in Figure 1c. On-chip cache memory is included to store data outputs of the first column of radix-4 butterflies for use as data inputs in the radix-4 butterfly operations of the second column to implement high-speed processing that is maximally execution-bound, as illustrated in Figure 2c.

Brief Description of the Drawings

Figure 1a is a simplified diagram of a conventional radix-2 butterfly operation;

Figure 1b is a simplified diagram of a conventional radix-4 butterfly operation;

Figure 1c is a simplified diagram of one embodiment of a quasi radix-16 butterfly operation according to the present invention;

Figures 2a-c are timing diagrams for the associated radix-n butterfly processors that illustrate input/output bandwidth and execution limitations;

Figure 3a is a simplified schematic diagram of a preferred embodiment of the quasi radix-16 butterfly hardware implementation according to the present invention;

Figure 3b is a simplified chart illustrating data flow in a quasi radix-16 butterfly processor according to the present invention;

Figures 4a and 4b are pictorial representations of the processing sequences for a conventional radix-4 processor and the quasi radix-16 processor of the present invention;

Figure 5 is a timing diagram illustrating the impact of data dependencies in a datapath on the processing speed attributable to the quasi radix-16 processor of the present invention;

Figure 6 is a chart illustrating the data processing for maximum execution utilization of a quasi radix-16 processor according to the present invention;

Figure 7a is a simplified diagram illustrating the data flow through an interleaved quasi-radix-16 scheme as illustrated in Figure 7b, according to the present invention;

Figure 7b is a simplified schematic diagram illustrating an interleaved quasi radix-16 butterfly operation with multiple twiddle inputs for operation as illustrated in Figure 4b according to the present invention;

Figure 7c is a chart illustrating the address orientations of twiddle inputs within on-chip memory in hexadecimal notation;

Figure 8 is a simplified diagram for performing a N-point Fast Fourier Transform with a quasi radix-16 butterfly of the present invention illustrating the generic notation;

Figure 9 is a block schematic diagram of the preferred embodiment of the circuit module radix-4 processor for operation in the quasi radix-16 processor according to the present invention;

Figure 10 is a schematic diagram of the memory and addressing circuitry associated with the quasi-radix 16 processor;

Figures 11a and 11b are charts illustrating the input memory storage scheme for data inputs;

Figures 12a-c are charts illustrating the coefficient memory storage scheme for twiddle input data;

Figures 13a and 13b are charts illustrating the cache memory storage scheme; and

Figures 14a-c are charts illustrating the output data memory scheme.

Description of the Preferred Embodiment

In general, a Fourier transform of data points representing a complex function includes an ascending series of terms, each including a coefficient and an angular velocity (or frequency) factor, where each term of the complex function may include real and imaginary quantities, in the form:

$$D_0 + D_1 \times W_1 + D_2 \times W_2 + D_3 \times W_3 \qquad \text{(Eq. 1)}$$

Thus, the higher-order terms of the complex function, in the general case, require four multiplications of the real and imaginary components per term, where D_n is data and W_n is a "twiddle" input such that

$$D_n = a+jb \qquad \text{(Eq. 2)}$$

$$W_n = c+jd \qquad \text{(Eq. 3)}$$

$$D_n \times W_n = (a+jb)(c+jd) =$$

$$axc + axjd + cxjb + jbxjd \quad \text{(Eq. 4)}$$

Radix-4 processing performs four multiplications per term, or twelve multiplications for these higher-order terms $D_1 \times W_1$ through $D_3 \times W_3$. The data inputs D_n and the twiddle inputs W_n are supplied at separate inputs for processing in selected sequence, as later described herein.

Referring now to Figure 1c, there is shown the quasi radix-16 butterfly which includes two columns of four radix-4 butterflies each (A to D and E to H), as shown in Figure 1c. The QR16 processor uses on-chip cache memory, later described herein, to store the 16 data outputs of the first column butterfly operations, for use in the second column butterfly operations. Figures 3a, 9 and 10 illustrate the hardware implementation of QR16 processor around a radix-4 butterfly execution unit. As can be seen from these figures, the QR16 processor is implemented by adding a limited amount of on-board memory 9, 11, 13, 15 to a radix-4 butterfly processor chip.

Each QR16 butterfly requires 16 input data points (D0-D15) and 15 twiddle factors (wl-wl5) as shown in Figure 1c, which takes 16 fetch cycles with 2 complex data input ports i.e., input data and twiddle inputs. Internally, eight

5   radix-4 butterfly operations (A to H in Figure 1c) are required. The first column uses the same data ordering as a 16-point radix-4 fast-Fourier transform (FFT) and all butterflies (A to D) have three common twiddles i.e., wl, w2, and w3. The second column of butterflies E through H require

10   digit reversed data ordering and 12 unique twiddles i.e., w4 to wl5. Without the QR16 butterfly, a 16 point radix-4 FFT would required 32 (2x16) data inputs, 24 (2x12) twiddle inputs, and 32 (2x16) data outputs, since the first column results would have to be stored in off-chip memory and be

15   brought in again on-chip to complete the second column.

With the QR16 butterfly scheme of the present invention, it is possible to achieve as much as twice the throughput of a conventional radix-4 butterfly based processor. The higher throughput is achieved by maximizing

20   the utilization of the datapath, as shown in Figure 2c. It should be noted in Figure 2c that each data fetch and output store e.g., A1* and A1*' respectively, actually takes four input/output (I/O) cycles. The 16 data points (A1*, B1*, C1* and D1*) brought on-chip are passed twice through the

25   datapath (Pass 1-1 and Pass 1-2) before being stored in external memory. Thus, for QR16 processor of the present invention, 16xTio = 16xTeu or Tio is equal to Teu and the

9

datapath is continuously busy, thus indicating an ideal I/O
balanced environment.

Referring now to Figure 4, the chart illustrates
the difference between a radix-4 butterfly sequence and a
5   QR16 butterfly sequence for 1K data points of a fast-Fourier
transform. Figure 4a shows the sequence of butterfly
operations for a 1K point FFT using only radix-4
butterflies. Each shaded rectangle in Figure 4 represents
one radix 4 butterfly operation. All the butterflies of a
10   single column are processed before the next column of
butterflies are processed. Thus, a 1K point FFT can be
performed in five columns by five radix-4 passes (1024 =
4x4x4x4x4) as shown in Figure 4a. In contrast, a 1K point
FFT can be performed in three columns by two QR16 butterfly
15   passes and one radix-4 pass (1024 = 16x16x4) as shown in
Figure 4b. In Figure 4b, the QR16 butterfly is shown as a
dashed line enclosing 8 radix-4 butterflies arranged in 2
columns. In Figure 4b, processing also proceeds column by
column except only three columns are needed because of the
20   higher radix used in the first two columns. The QR16
butterfly essentially allows two radix-4 columns to be
processed at a time before the next stage. The QR16
processor advantageously includes on chip cache memory 15 for
storing real and imaginary data needed for processing the two
25   radix-4 columns in the QR16 butterfly. The present
invention advantageously uses on chip memory to rearrange
radix-4 butterflies to exploit data locality to increase the

10

speed of computation. In addition to cache memory, input
and output memories are included, as shown in Figure 3a. The
QR16 embodiment of the present invention includes an inherent
'pipeline' latency problem, when implemented on a pipelined
radix-4 datapath. In accordance with this invention, this
latency can be eliminated by interleaving the QR16
computations of two 16-point data blocks in the pipelined
datapath, as later described herein. Table 1, below,
illustrates the increases in processing speed that are
achieved using the QR16 of the present invention, especially
when operated in radix-16 mode only.

Table 1

FFT Size	Passes	FFT Execution Time QR16	Only Radix-4	Speed Improvement Factor
64	4x16	2.56us	3.84us	1.5
256	16x16	10.24us	20.48us	2.0
1K	4x16x16	61.44us	102.4 us	1.67
4K	16x16x16	245.76us	491.52us	2.0
16K	4x16x16x16	1.31ms	2.29ms	1.75
64K	16x16x16x16	5.24ms	10.48ms	2.0

Assuming: Tio = Ten = 20ns

The Quasi-radix 16 Butterfly Implementation

Referring now to Figure 3a and 10, in order to
implement QR16 using a radix-4 module 7, a limited amount of
on-board memory has to be included with this module on the
same circuit chip. Memory structures 9, 11, 13 are placed at
the input and output of the datapath. In addition, a cache
memory 15 is also required. For complex data, the memory
word width should allow for real and imaginary data
components. The input and output memories 11, 13 are used

11

for data buffering while the cache memory 15 is used to store the intermediate results of column 1 of the QR16 butterfly processing, as illustrated in Figure 1c. Figure 3b shows the basic data flow through the memories and the datapath. A more detailed version will be discussed later herein. Input data points are stored in the input memory 11 and the twiddles are stored in the coefficient memory 9. Then, the datapath operates on this data to accomplish the first column of QR16 butterfly processing. The results of the first column butterflies are stored in the cache memory 15 and once all the butterflies of column 1 are complete, the cache memory 15 is read to perform the second column butterflies. The final results of the second column of a QR16 butterfly are stored in the output memory 13 for transfer to off-chip memory.

Typically, multiple stage pipelined data paths are used to implement high performance FFT processors, e.g., a pipeline stage for the multiplier section, a pipeline stage for the adder section, a pipeline stage for the ALU section, and the like. In the QR16 butterfly of the present invention, there exists a data dependency between the first column and the second column of butterfly processing. The results of all first column butterflies must be available before the second column calculations can begin, i.e., with reference to Figure 1c, the results of butterfly D must be stored in the cache memory 15 prior to the execution of butterfly E. This data dependency becomes an acute problem

in pipelined data paths where the results of a particular
operation are not available until a specified number of clock
cycles later due to the latency of the datapath pipeline. In
the QR16, this would impose a certain number of dead (or

5   no-op) cycles into the pipeline after the last butterfly of
the first column (butterfly D) and before the first butterfly
of the second column (butterfly E) to avoid this data
dependency.

This data dependency can be understood clearly from

10   the timing diagram of Figure 5 which shows nine timing
patterns corresponding to the memory read and write functions
and the execution unit, where each of these waveforms is
defined below:

     (a) Input Memory Write - Input data from I/O pins

15   written in Input Data Memory 11;

     (b) Coeff Memory Write - Twiddle data from I/O pins
       written to Coefficient Memory 9;

     (c) Coeff Memory Read - Input data transfer to
       Execution Unit 7;

20   (d) Input memory read - Input data transfer to
       Execution Unit 7;

     (e) Execute - Execution Unit activity;

     (f) Cache Write - Execution unit result data
       written to Cache Memory 15;

25   (g) Cache Read - Cache data transfer to Execution
       Unit 7;

13

(h) Output Memory Write - Execution unit result

data written to Output Data Memory 13; and

(i) Output Memory Read - Output data read to I/O

pins.

Referring now to the graph of Figure 5, A and B

refer to two different QR16 butterfly data blocks. A1 and A2

refer to the first and second columns of the A block

respectively. Similarly, B1 and B2 refer to the first and

second columns of the B block, as illustrated in Figure 7b.

In the timing waveforms of Figure 5, the shaded areas on a

waveform indicate no activity for that function for the

specified duration, whereas unshaded areas indicate

processing activity.

The timing diagram in Figure 5 assumes Teu = Tio,

i.e., pipeline cycle time is equal to I/O cycle time. First,

the input data for block A is written to the input memory 11;

this takes sixteen (Tio or Teu) cycles to bring in the

sixteen data points of block A and is shown on the Input

Memory Write pattern (a). Similarly, the twiddles for block

A are also written to the coefficient memory 9 in sixteen

cycles. Note that the Coeff Memory Write (b) is delayed 16

by eight cycles with respect to the Input Memory Write (a).

This situation arises because the Coeff Memory Read A1 only

empties three memory locations. As will be shown later, the

twiddle sequencing requires retransmission of three twiddles

for the first column butterflies. Therefore, to keep the

input 11 and coefficient 9 Memory of the same depth, the Coeff Memory Write (b) is delayed 16 by eight cycles with respect to the Input Memory Write (a). Two cycles prior to the completion of Input Memory Write (a), the execution unit 7 accesses the input data memory 11 (Input Memory Read (d)) and the coefficient memory 9 (Coeff Memory Read (c)) to get the data for the first column butterflies of block A. Note that the read process only takes eight cycles 16 since the read bandwidth is twice the write bandwidth for these memories 9, 11, as later described. Therefore, two cycles are required to fetch the data for one radix-4 butterfly from the memories 9, 11. Hence, the execution unit activity can begin two cycles after the Coeff (c) and Input (d) Memory Read, as shown on the Execute waveform (e). Each column of a QR16 butterfly requires that eight cycles be initiated since there are four butterflies in each column. Note that the transitions shown on the Execute waveform (e) only indicate the beginning of a new column of butterflies and not the total execution time of a column through the pipeline. The radix-4 datapath on which the QR16 of the present invention is implemented has an execution latency of six cycles 18. Therefore, the results of A1 start getting written into the cache memory six cycles 18 after the execution (A1) is begun. The Cache Read (A1/A2) (g) is initiated as soon as all the results of A1 are available. The twiddles for column 2 are accessed at the same time (Coeff Memory Read A2) (c) from the Coefficient Memory 9. Execution for column 2, i.e.,

A2 can be started two cycles later. Six cycles 20 after the
start of A2 execution, the results of these computations can
begin to be written (h) into the output memory 13, and the
data in the output memory 13 can be read (i) to the I/O pins

5   (Output Memory Read). For the output memory 13, the write
bandwidth is twice the read bandwidth, as described later
herein, and thus the results of a column of butterflies are
written in eight cycles and read in sixteen cycles. The
sequence of operations described above is repeated for block

10   B and every other block thereafter. It should be noted,
however, that the shaded areas in the Execute timing waveform
(e) indicate dead time periods 22 attributable to this data
dependency in the processing through the QR16. Specifically,
for this datapath, a dead time period 24 consisting of seven

15   Teu (or Tio) cycles would be introduced into the pipeline
between the A1 and A2 execution.

However, in accordance with the present invention
this latency can be eliminated by interleaving the QR16
computations of two 16-point blocks in the datapath. To

20   accomplish this, twice as large on-board memories are used to
store two 16-point data blocks on-chip at the same time, as
illustrated in the timing diagram of Figure 6 which shows the
timing diagram for two interleaved QR16 butterfly
computations. The timing waveforms shown in Figure 6 have

25   the same definitions as in Figure 5, and A and B again refer
to two QR16 butterfly data blocks. The input data for the
two QR16 butterfly data blocks (A and B) is written (a) to

the input data memory 11 sequentially, and this takes 32

(2x16) cycles. However, twiddle data for both block A and B

is written (b) to the coefficient memory 9 in a scrambled

fashion, which will be explained later, and is again delayed

26 by sixteen cycles with respect to the Input Memory Write

5   (a). Once the input data for a pair of QR16 blocks is

available in the input data memory 11, the execution unit 7

reads the input data memory 11 to start the first column

block A computations (e) (A1). As soon as the data for the

last butterfly of A1 has been read, the first column of block

10   B computations are initiated. The sequence of operations

just described for interleaved QR16 butterflies is

illustrated in Figure 7a, and a more detailed version of the

computation is illustrated in Figure 7b. It should be noted

in Figure 6 that as soon as B1 computations are complete, the

15   results of A1 are already available for fetching from the

cache memory 15 to compute A2. As this computation is

completed, the results of B1 are already available in the

cache memory 15 for computing B2. Finally, this process can

be repeated for the next pair of QR16 data blocks, and so

20   on. This interleaving scheme in accordance with the present

invention requires 32 storage locations in all the on-board

memories 9, 11, 13 and 15. As can be seen from the Execute

timing waveform (e) in Figure 6, there are now no no-op or

shaded areas 26 on this waveform, thus indicating a

25   continuously busy datapath. Figure 7c shows the twiddle or

coefficient sequence for this interleaved scheme, as

explained below in detail later herein in connection with
the addressing for the coefficient memory 9.

Having defined the basic memory requirements for
QR16 and having discussed the interleaved QR16 scheme, the
details of each of the on-board memories will now be
described. The cache memory 15 is a conventional
double-buffered structure or dual-port memory to allow
concurrent read/write for the pipelined datapath. As
indicated in Figure 6, to keep the datapath pipeline
continuously busy with the interleaving scheme, the cache
memory 15 must be written with the results of B1 and at the
same time the results of A1 need to be read. The cache
memory 15 is preferably comprised of four banks 24 bits wide
and 16 words deep. Therefore, the cache memory 15 may have
32 complex data words (48 bits) locations or 64 real data
word (24 bits) locations. The word width is governed
directly by the data in the datapath on which the QR16 is
implemented. The cache memory 15 must be able to supply and
accept two complex words every clock cycle. The cache memory
addressing can be accomplished for example, by a four-bit
counter and a few multiplexers (mux). Some conventional
counter and mux logic supports sequential data ordering as
well as conventional digit-reversed data ordering. As
mentioned above, the results of the first column of a QR16
butterfly are stored in digit-reversed order and read in
sequential order for the second column calculations.

Input data memory 11 and output data memory 13 placed at the input and output of the datapath, respectively, are of conventional design and used for data buffering and concurrent read/write operation. Both the input and output

5   memories 11, 13 are also four banks 24 bits wide and 16 words deep having 32 complex data words (48 bits) locations or 64 real data word (24 bits) locations. The memory buffering structures are used here for the reasons that there is a speed gap between slow I/O and a fast datapath, and that

10   there is a need for convenient simultaneous read/write operations. On the proposed datapath, a radix-4 butterfly can be executed every two pipeline cycles. This implies that two data quantities must be supplied every pipeline cycle (2 data per cycle x 2 cycles = 4 data values per radix-4

15   butterfly) if the data and twiddle have separated dedicated I/O ports. However, it is generally not possible to supply two data values every cycle with just one data and one twiddle port if the I/O cycle time is the same as the pipeline cycle time. This is the speed gap that exists which

20   is eliminated by using memory structures 9, 11, 13 at the input and output of the datapath. The timing shown in Figure 6 assumes one common clock between I/O and execution unit in order to minimize the complexity of the control structure. With a common clock, it becomes necessary to have the read

25   bandwidth of the input memory 11 be twice the write bandwidth and vice versa for the output memory 13. The timing shown therefore avoids boundary conditions in the memories for ease

of implementation. Even if different clocks are used for the I/O and the datapath, the use of memory structures at the input and output eases the interface problem between external circuitry and the internal datapath.

In addition to the input and output memories 11, 13, there is also a coefficient memory 9 which is used for storing the twiddle data, as shown in Figure 3a. The coefficient memory 9 is preferably comprised of four banks 24 bits wide and 20 words deep. Thus, the coefficient memory 9 may have 40 complex data words (48 bits) locations or 80 real data word (24 bits) locations. However, only 30 locations are used to store two sets of 15 twiddles corresponding to the pair of QR16 data blocks resident in the input memory 11 for the interleaving scheme. The coefficient memory 9 structure is otherwise similar to the input and output memories 11, 13 previously described. The coefficient memory 9 also has a read bandwidth twice that of the write bandwidth, and is of conventional configuration for supporting non-sequential read addressing. It should be noted that three twiddle factors, W1, W2, and W3, are shared across all radix-4 butterflies in the first column of data block A, as shown in Figure 7b. For the second column butterflies of data block A, 12 unique twiddles, W4 to W15, are read from the coefficient memory 9 sequentially. In the interleaving scheme according to the present invention as illustrated in Figure 7c, W1 to W3 are read 4 times for the first column calculation of data block A (A1) and then W16 to

W18 are read 4 times for the first column calculation of data block B (B1). Then W4 to W15 are read sequentially for second column calculations of data block A (A2) and W19 to W30 are read similarly for second column calculations of data block A (A2) and W19 and W30 are read sequentially for second column of data block B (B2). The required read sequence for the twiddles in the interleaving scheme of the present invention is shown more clearly in Figure 7c which indicates the address locations of the twiddle inputs (numbered in hexadecimal notation). The coefficient memory 9 only supports read-retransmission from the required locations, and the combined sequence between the first 15 and the second 15 twiddles is supported by an external address generator 21, as illustrated in Figure 10.

In the QR16 according to the present invention, two levels of address generation are required. First, a given size FFT must be composed of QR16 blocks and the appropriate number of passes of radix-16, radix-4 and radix-2 must be calculated. The equations for data and twiddle sequencing to perform an N point FFT by QR16 are shown in Figure 8. The data and twiddle addressing at this block level, where a pair of 16-point data blocks are brought from external memory (MEM) 32, 34 and 36, is done by an off-chip conventional address generator (AG) 21, 23, 25, as shown in Figure 10. As previously described, the external AG 21, 23, 25 must combine the twiddles of two QR16 blocks in the manner shown in Figure 7c. At the second level, the address generation is required

within the pair of data blocks and this, as previously
discussed, is handled on-chip by conventional addressing
circuitry associated with each on-board memory 9, 11, 13 and
15.

5   Referring now to Figure 9, there is shown a
schematic diagram of processor circuitry according to the
present invention for operation in the quasi-radix-16
embodiment. This circuitry and the datapath therethrough can
accomplish a radix-4 complex butterfly every two cycles using

10   6 multipliers 27, 29, 31, 33, 35 and 37, and with 3 stages of
adders 39-42, 44-47 and 49-52, to permit a 16-cycle QR16
butterfly processing of data and twiddles applied to input
registers 53-66. The data bandwidth required by this
datapath for the QR16 butterfly is achieved by doubling the

15   read/write bandwidth of the on-board memories. Thus, on each
execution cycle, the on-board memories can supply/accept two
sets of two complex numbers. Thus, with reference to the Eq.
1, above, the data for processing successively higher-order
terms is arranged generally from left to right in that the D_0

20   data inputs are supplied via data registers 53, 54, the D_1
data inputs are supplied via data registers 55, 56, the D_2
data inputs are supplied via data registers 57, 58 and the D_3
data inputs are supplied via data registers 59, 60.
Similarly, the W_1 twiddle inputs are supplied via data

25   registers 61, 62, the W2 twiddle inputs are supplied via data
registers 63, 64, and the W3 twiddle inputs are supplied via
data registers 65, 66. The groups of multiplexers 67, 68 and

69, 70 and 71, 72 are cross connected to facilitate the
complex cross-product multiplication in the respective
multipliers during alternate cycles to yield products that
are stored in registers 74-81. The initial products are
arithmetically combined in adder/subtractor units 39-42 and
stored in the registers 82-85. The outputs of registers
82-85 again combined in a second stage of adder/subtractor
units 44-47 and stored in data registers 87-94. The data in
registers 87-94 is supplied via multiplexers 95-96 to the
arithmetic logic units 49-52. Registers 98-102 are coupled
to receive and store the results from the arithmetic logic
units 49-52. The outputs of these registers 98-102 are
coupled to a data formatter multiplexer 104 that organizes
the data from the registers for output to the memories in the
desired 24 or 48 bit format. Thus, the complex data for D_1W_1
is processed in multipliers 27, 29 during first and second
cycles, the complex data for D_2W_2 is processed in multipliers
31, 33 during first and second cycles, and the complex data
for D_3W_3 is processed in multipliers 35, 37 during first and
second cycles. Ultimately, the processed complex outputs
(say, from first column processing appear at outputs 106, 107
for D0 in one cycle, and at outputs 108, 109 for D_1 in the
one cycle, and appear at outputs 106, 107 for D2 in the
second cycle, and at outputs 108, 109 for D_3 in the second
cycle. With the aid of cache memory 15 on-chip, these
outputs from first-column processing are stored to provide
the data inputs for second-column processing according to the

quasi radix-16, as previously described. This circuit also
supports radix-2 and radix-4 butterfly processing
operations. A radix-2 butterfly can be done in a single
cycle in this circuitry, but I/O only supplies half of the
required data in one cycle, which therefore limits the
radix-2 processing to 2 cycles. Radix-4 butterfly processing
can be executed in 2 cycles, but for the same reason, it
takes 4 cycles to read the data. In this circuitry, the QR16
butterfly is the only operation that achieves 100% execution
utilization, as shown in Figure 6. More specifically, the
QR16 according to the present invention is most efficient if
the FFT size is an integral power of 16, for example 256 (16
x 16), 4K (16 x 16 x 16), as illustrated in Table 1 above,
but is also capable of speeding up the computation by 30 to
50% for other FFT sizes. Assuming a 20 nanoseconds I/O rate
and a 40 nanoseconds radix-4 butterfly, the QR16 according to
the present invention performs a 1K FFT in 61.4 microseconds
compared with 102.4 microseconds if only radix-4 butterfly
passes were used on the same circuitry. Performance
improvements due to the QR16 according to the present
invention for other FFT sizes are also tabulated in Table 1,
above.

Referring now to Figures 11-14, there is shown a
pictorial representation of the preferred memory storage
scheme for the coefficient, input, output and cache memories
9, 11, 13 and 15. Each of the memories 9, 11, 13 and 15
comprise a left bank of complex memories (A+jB) and a right

bank of complex memories (C+jD). The memories 9, 11, 13 and 15 preferably have 20 row addresses designated in the top row of each chart. Therefore, at each address one word may be written the left bank and another word may be written in the right bank of complex memory. The charts of Figures 11-14 detail the write and read sequences for words written the respective memories. For example, as shown in Figure 11, the write sequence for Radix 2, 4 or 16 (a) in the input data memory 11 writes word 0 at the left bank (A+jB) of address 0, word 1 at the right bank (C+jD) of address 0, and word 2 at the left bank (A+jB) of address 1. Words 3-1f are similarly written according to the write sequence indicated in Figure 11a. Exemplary schemes for reading and writing data in the memories 9, 11, 13 and 15 are shown in remaining charts of Figures 11-14. The data is preferably retrieved and stored in the memories 9, 11, 13 and 15 with address sequences ordered linearly or in some form of reverse addressing. As noted above, the data in the Coefficient Memory 9, may be ordered in a scrambled fashion to permit 100% use of the execution unit 7 as described above with reference to Figure 6.

25

CLAIMS

1.   A processor for performing fast Fourier transforms, said processor comprising:

an input memory having inputs and outputs for storing data;

an output memory having inputs and outputs for storing data;

a cache memory having inputs and outputs for storing data;

a coefficient memory having inputs and outputs for storing data; and

an execution unit having inputs and outputs for performing logical and arithmetic operations on real and imaginary numbers, the outputs of said execution unit coupled to the inputs of the output memory and the cache memory, the inputs of said execution unit coupled to the outputs of the input memory, the outputs of the coefficient memory and the outputs of the cache memory.

2.   The processor of claim 1, wherein the input memory, the output memory, the cache memory and the coefficient memory are 32 words deep.

3.   The processor of claim 1, wherein the cache memory can supply and accept two words every clock cycle.

4.   The processor of claim 1, wherein the input memory and the coefficient memory have a read bandwidth twice the write bandwidth, and the output memory has a write bandwidth twice the read bandwidth.

5.   The processor of claim 1, wherein the execution further comprises:

a plurality of input registers having inputs and output for storing data;

a plurality of multipliers having inputs and outputs for performing cross product multiplication, the inputs of said plurality of multipliers coupled to the outputs of said registers;

a plurality of arithmetic logic units having an inputs and outputs for performing logical and arithmetic operations on the data input, the inputs of said first stage coupled to the outputs of the arithmetic logic units; and

a multiplexer having inputs and outputs for formatting the data input, the inputs of said multiplexer coupled to the outputs of the plurality of multiplexers.

6.   The processor of claim 5, wherein plurality of arithmetic logic units comprise:

a first stage of arithmetic logic units having an inputs and outputs for generating the sum or difference of data input, the inputs of said first stage coupled to the outputs of the multipliers;

a second stage of arithmetic logic units having an inputs and outputs for generating sums and differences, the inputs of said second stage coupled to the outputs of the first stage; and

a third stage of arithmetic logic units having an inputs and outputs for performing logical and arithmetic

operations, the inputs of said third stage coupled to the outputs of the second stage.

7.   The processor of claim 1, wherein the input memory, the output memory, the cache memory, the coefficient memory and the execution unit art constructed on a single chip.

8.   The processor of claim 1, wherein the execution unit is a radix-4 processor.

9.   The processor of claim 1, wherein the execution unit, input memory, output memory, cache memory and coefficient memory are constructed on a single chip.

10.   A method for performing fast Fourier transforms on a quasi radix-16 processor having an execution unit, input memory, output memory, coefficient memory and cache memory, said method comprising the steps of:

storing data in the input memory and coefficient memory;

reading data from the input memory and coefficient memory;

performing calculations on the data from input memory and coefficient memory with the execution unit;

storing the output of the execution unit in cache memory;

reading data from cache memory;

performing calculations on the data from cache memory, input memory and coefficient memory with the execution unit; and

storing the output of the execution unit in output

memory.

Figure 1b- A Radix-4 Butterfly

Figure 1a- A Radix-2 Butterfly
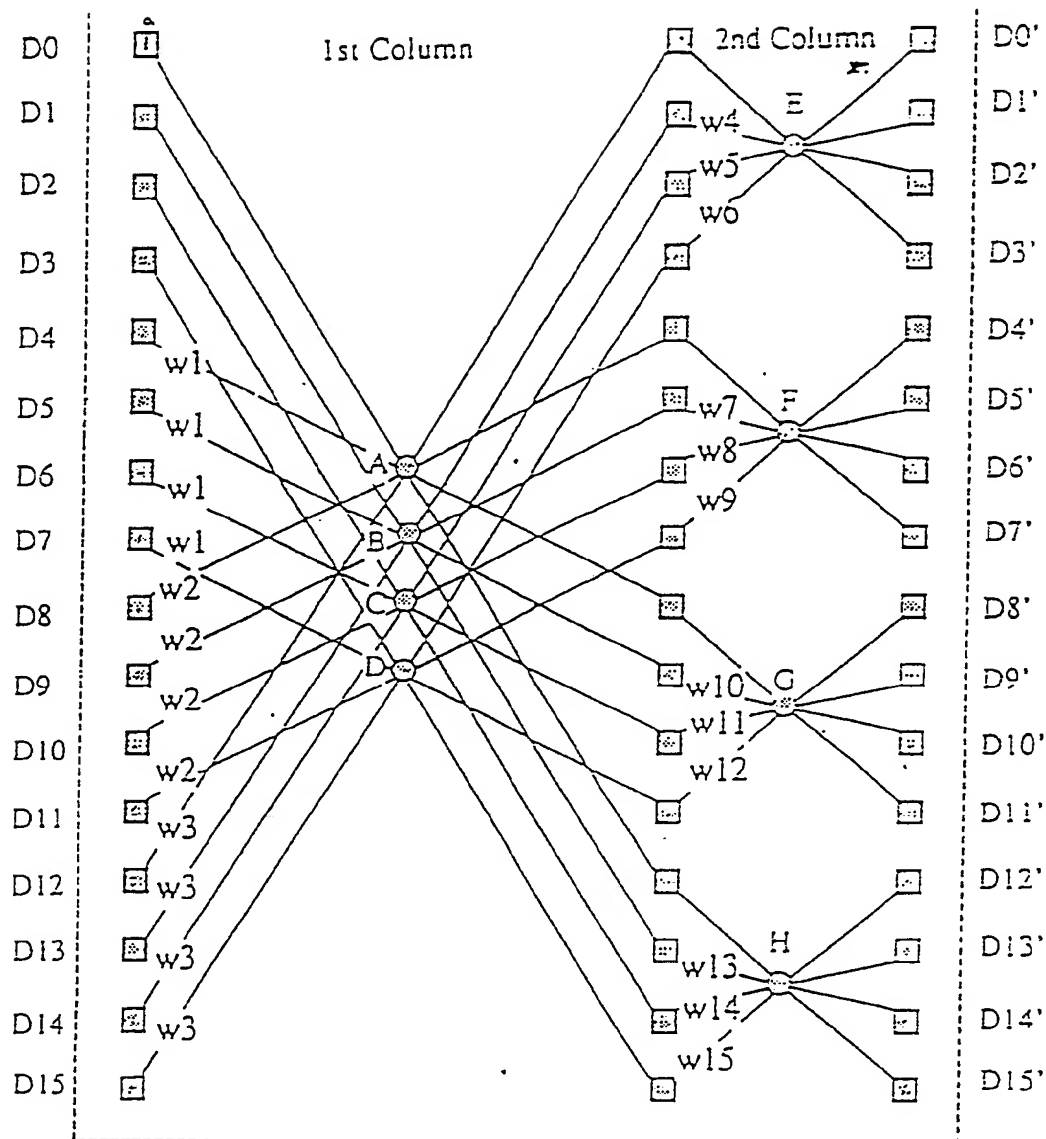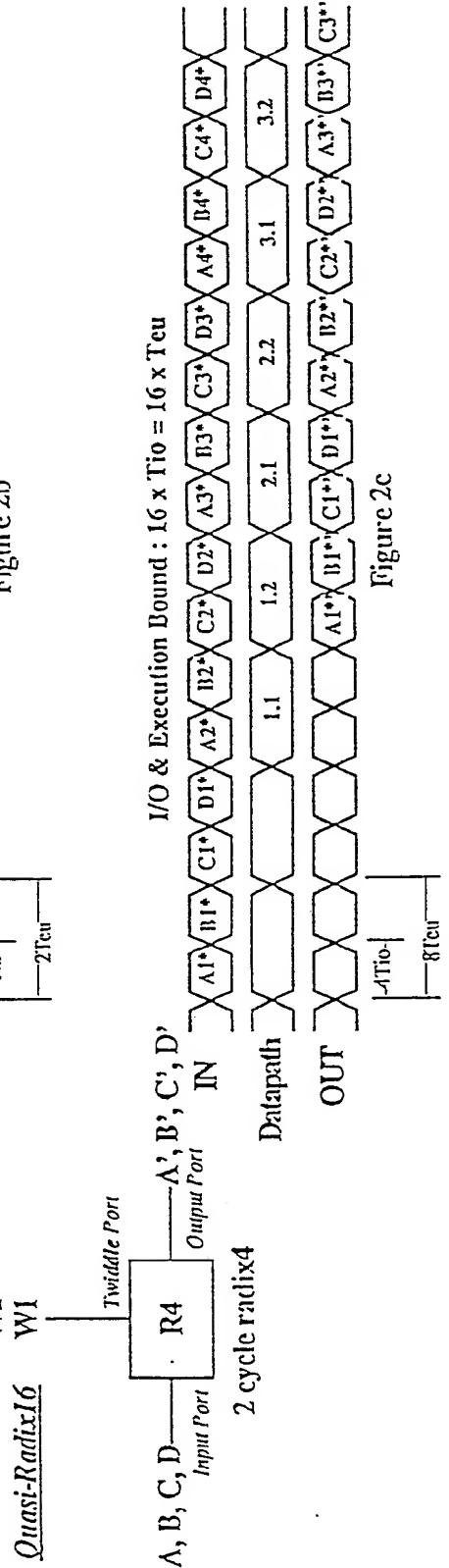
Figure 1c

3 / 16

I/O Bandwidth Limitation for Radix-2 and Radix-4

(Theoretical Timing Diagrams)

Radix2

A, B — Input Port → R2 → Output Port — A', B'

W — Twiddle Port

2 cycle radix2

I/O Bound : Tio = Teu (would be Execution bound if 4 x Tio = 2 x Teu)

IN: A1 B1 A2 B2 A3 B3 A4 B4 A5 B5 A6 B6 A7 B7 A8 B8

Datapath: 1,2   3,4   5,6

OUT: A1' B1' A2' B2' A3' B3' A4' B4' A5' B5' A6'

|Tio|

2Teu

Figure 2a

Radix4

A, B, C, D — Input Port → R4 → Output Port — A', B', C', D'

W3 W2 W1 — Twiddle Port

4 cycle radix4

I/O Bound : Tio = Teu (would be Execution bound if 4 x Tio = 2 x Teu)

IN: A1 B1 C1 D1 A2 B2 C2 D2 A3 B3 C3 D3 A4 B4 C4 D4

Datapath: 1   2   3   4

OUT: A1' B1' C1' D1' A2' B2' C2' D2' A3' B3' C3'

|Tio|

2Teu

Figure 2b

Quasi-Radix16

A, B, C, D — Input Port → R4 → Output Port — A', B', C', D'

W3 W2 W1 — Twiddle Port

2 cycle radix4

I/O & Execution Bound : 16 x Tio = 16 x Teu

IN: A1* B1* C1* D1* A2* B2* C2* D2* A3* B3* C3* D3* A4* B4* C4* D4*

Datapath: 1.1 1.2 2.1 2.2 3.1 3.2

OUT: A1** B1** C1** D1** A2** B2** C2** D2** A3** B3** C3**

|Tio|

8Teu

Figure 2c

4 / 16

Figure 3b
Basic Data Flow
in QR16
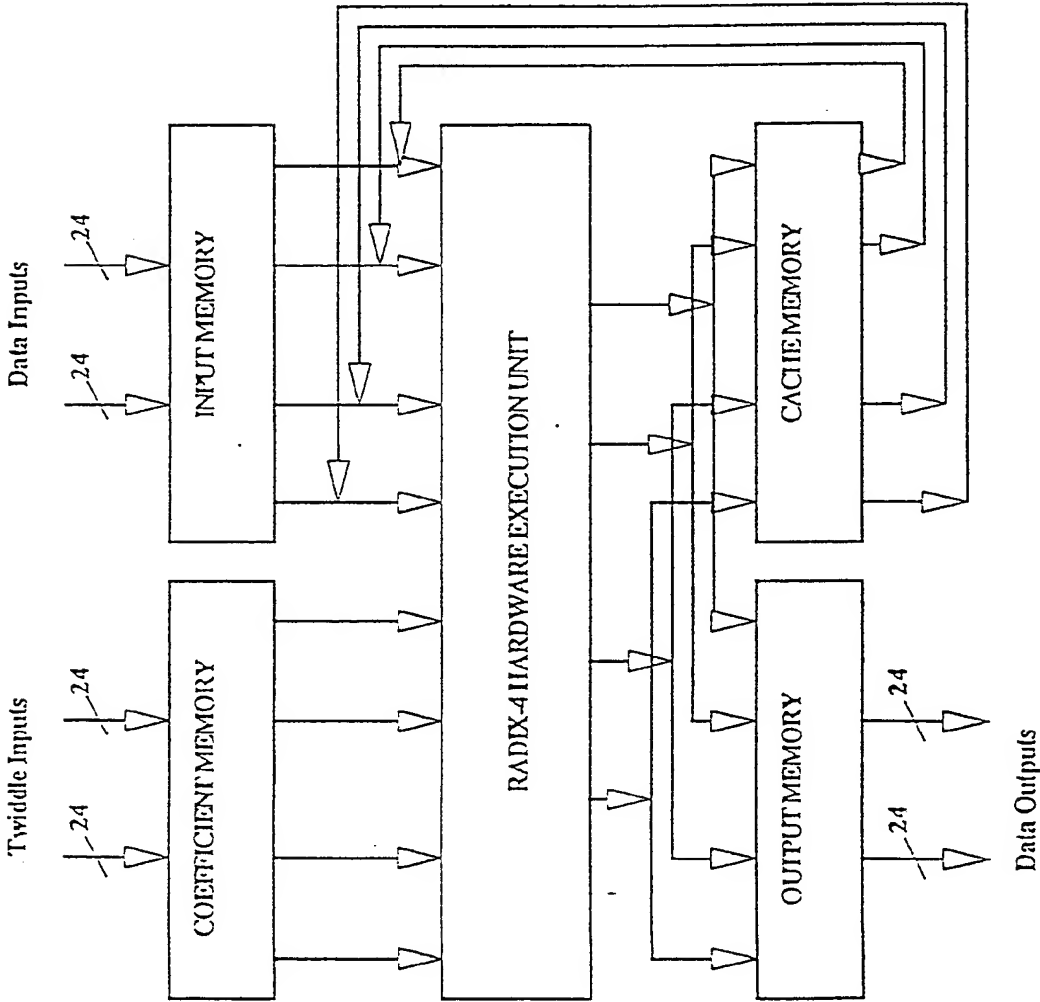
Figure 3a - Quasi Radix-16 Butterfly Hardware Implementation

5 / 16

1024 pt FFT

By Radix-4

All butterflies in one stage are processed before next stage is begun.

By Quasi Radix-16

Here the butterflies are arranged such that 2 passes of Radix-4 can be processed before off chip memory needs to be accessed. Also known as Two-At-A-Time Flowgraphs.

Figure 4 - Differences in Radix-4 & QR16 Butterfly Sequence for 1K FFT
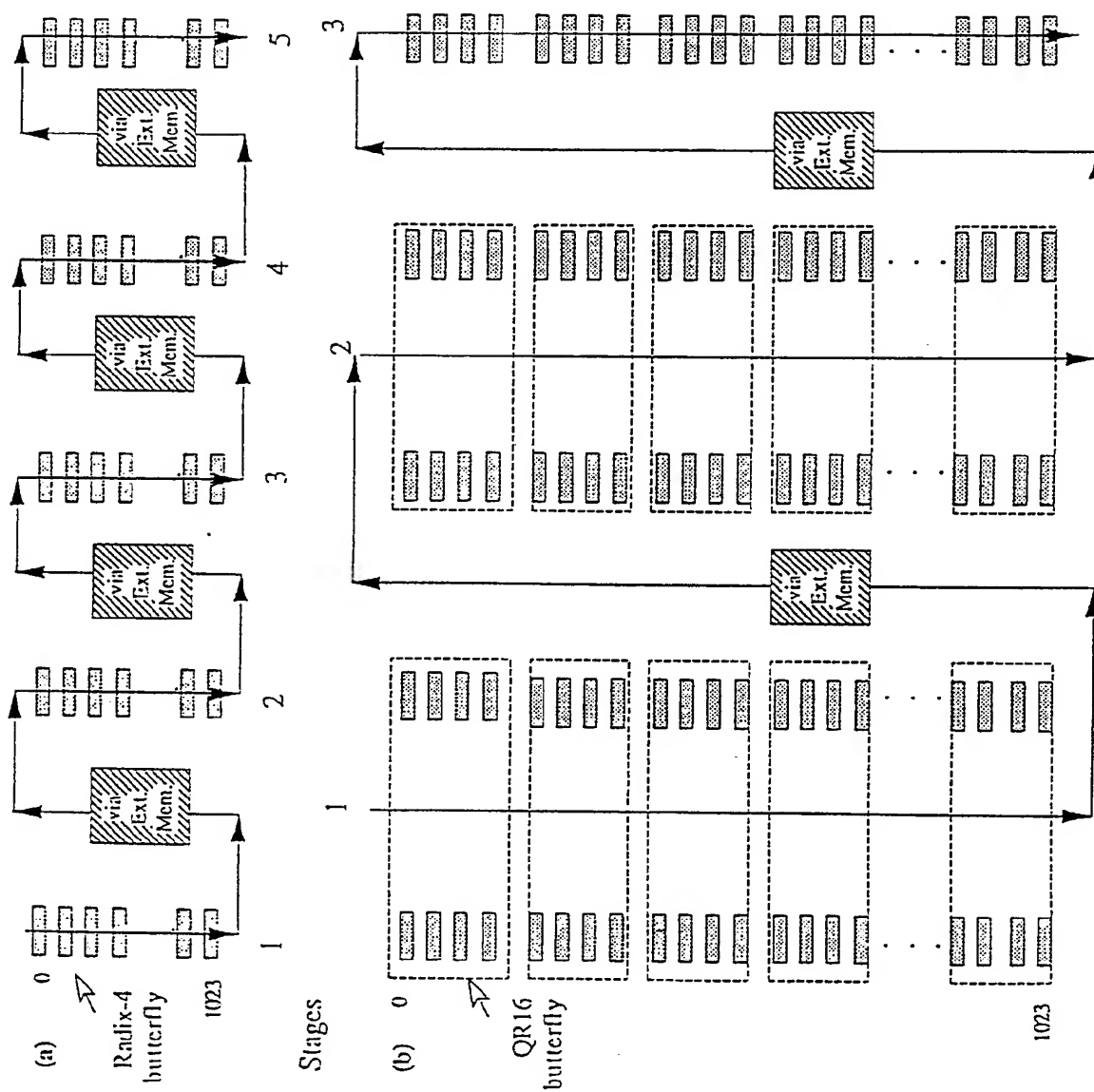
Figure 5 - Cache Memories & FIFOs Timing Diagram 1
Data Dependency in QR16 Butterfly
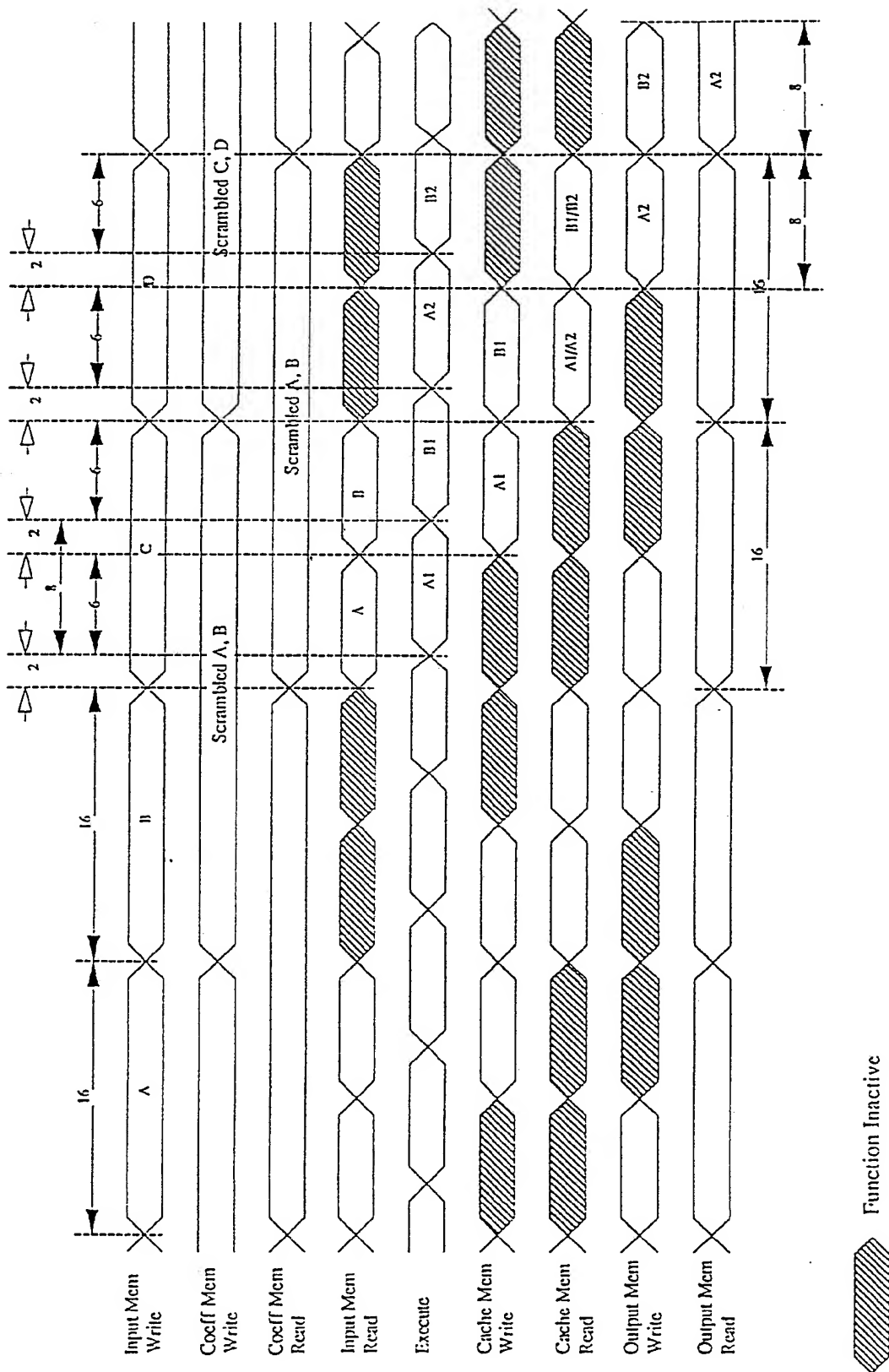
Figure 6 - Cache Memories & FIFOs Timing Diagram 2
Interleaved QR16 Butterflies

8 / 16
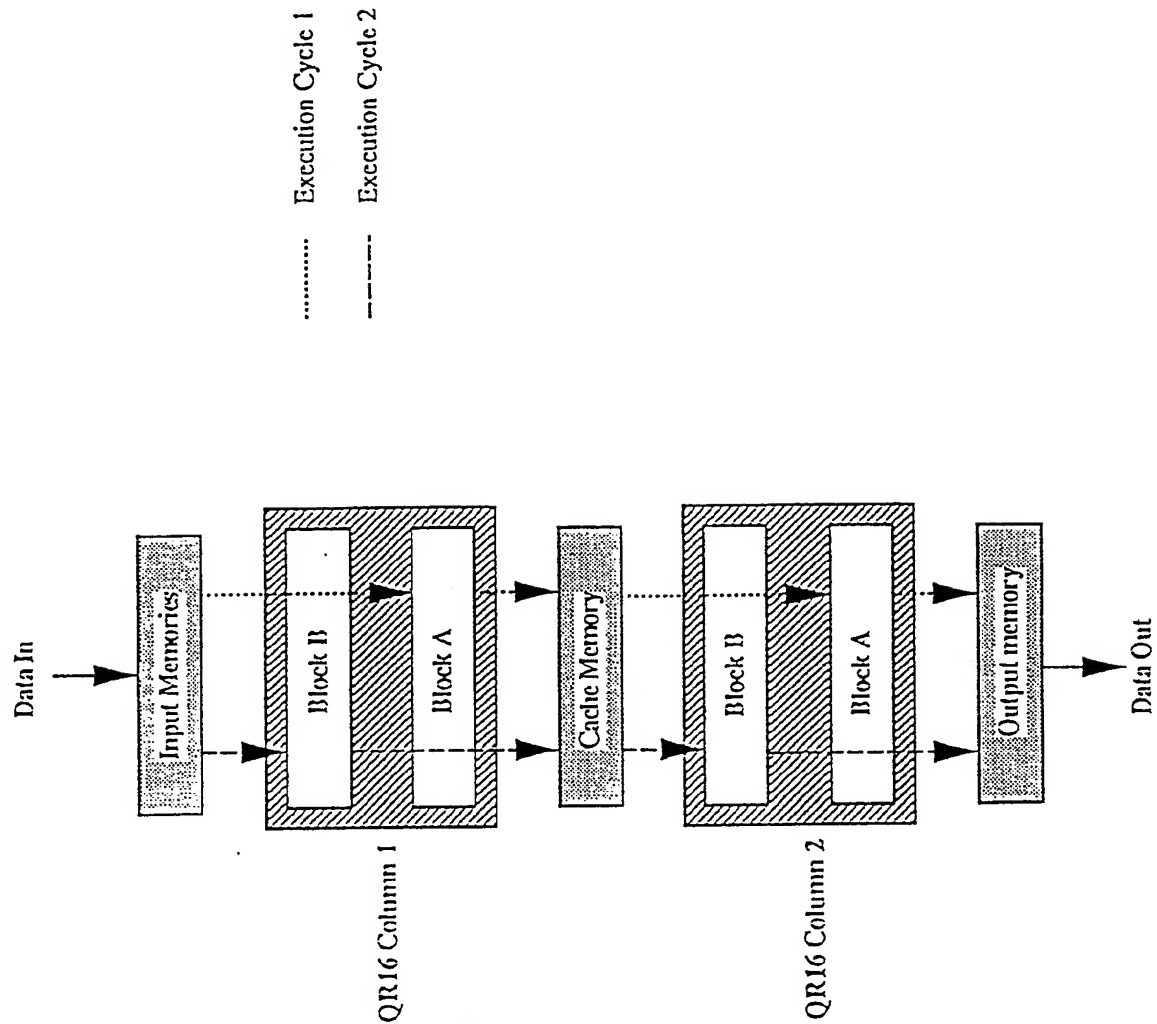
Figure 7a
Data Flow in
Interleaved QR16
Scheme

9 / 16
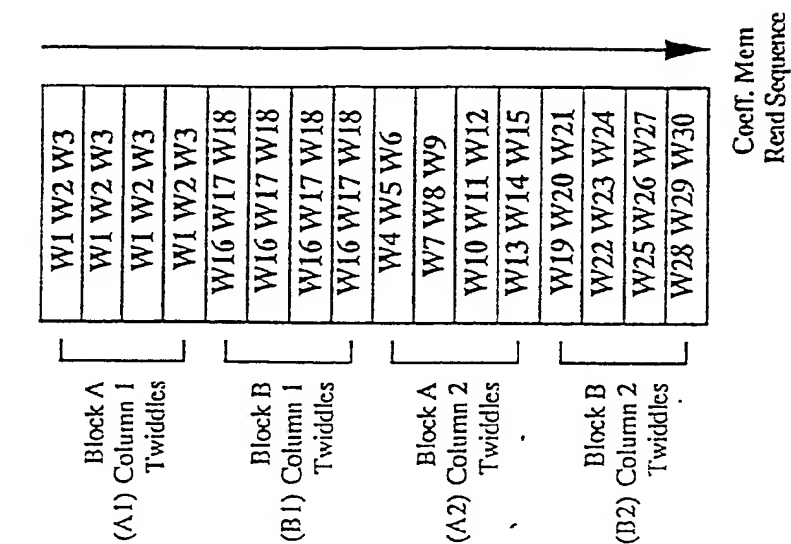
Figure 7c
Twiddle
Sequencing
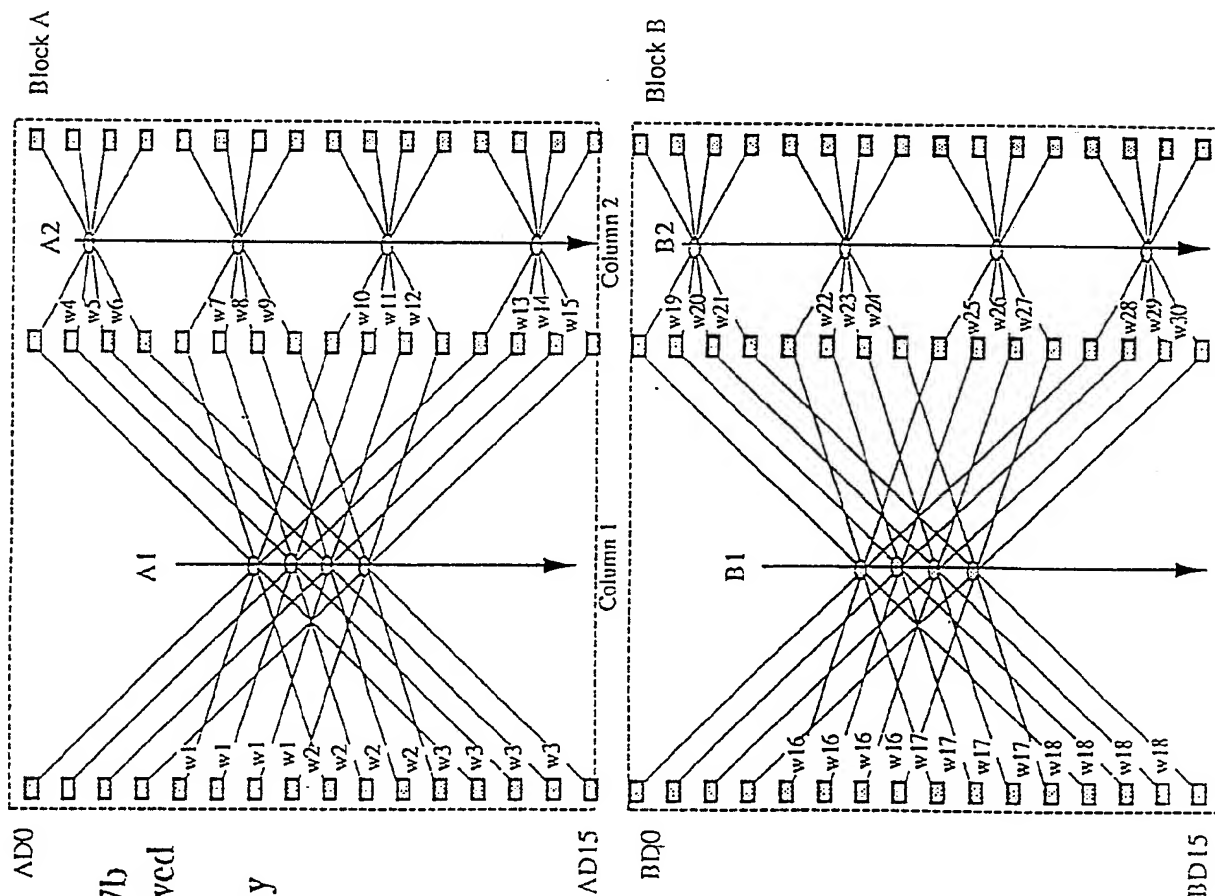
Figure 7b
Interleaved
QR16
Butterfly
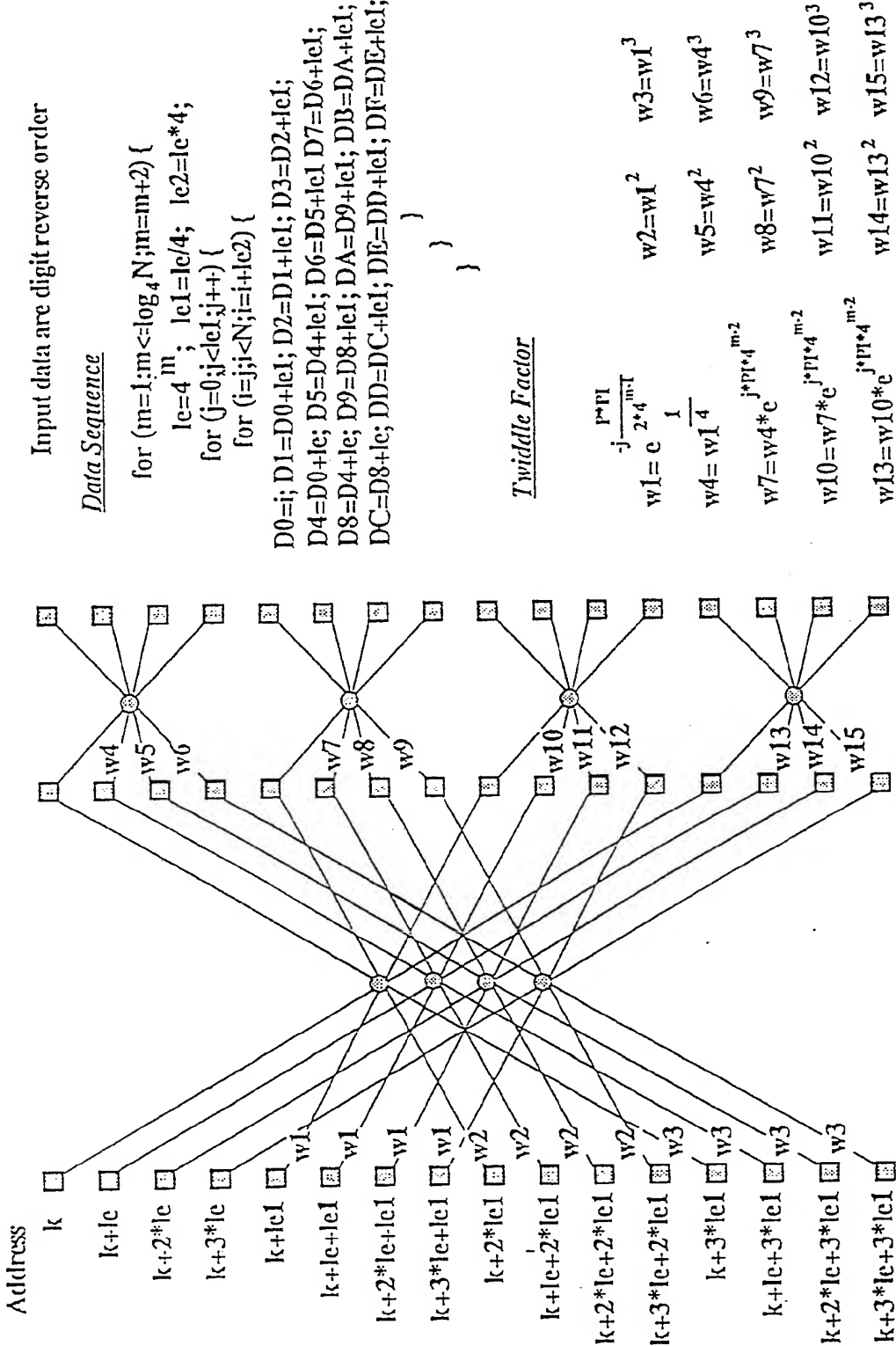
Input data are digit reverse order

Data Sequence

for (m=1;m<=log₄ N;m=m+2) {

    le=4^m;   le1=le/4;   le2=le*4;

    for (j=0;j<le1;j++) {

        for (i=j;i<N;i=i+le2) {

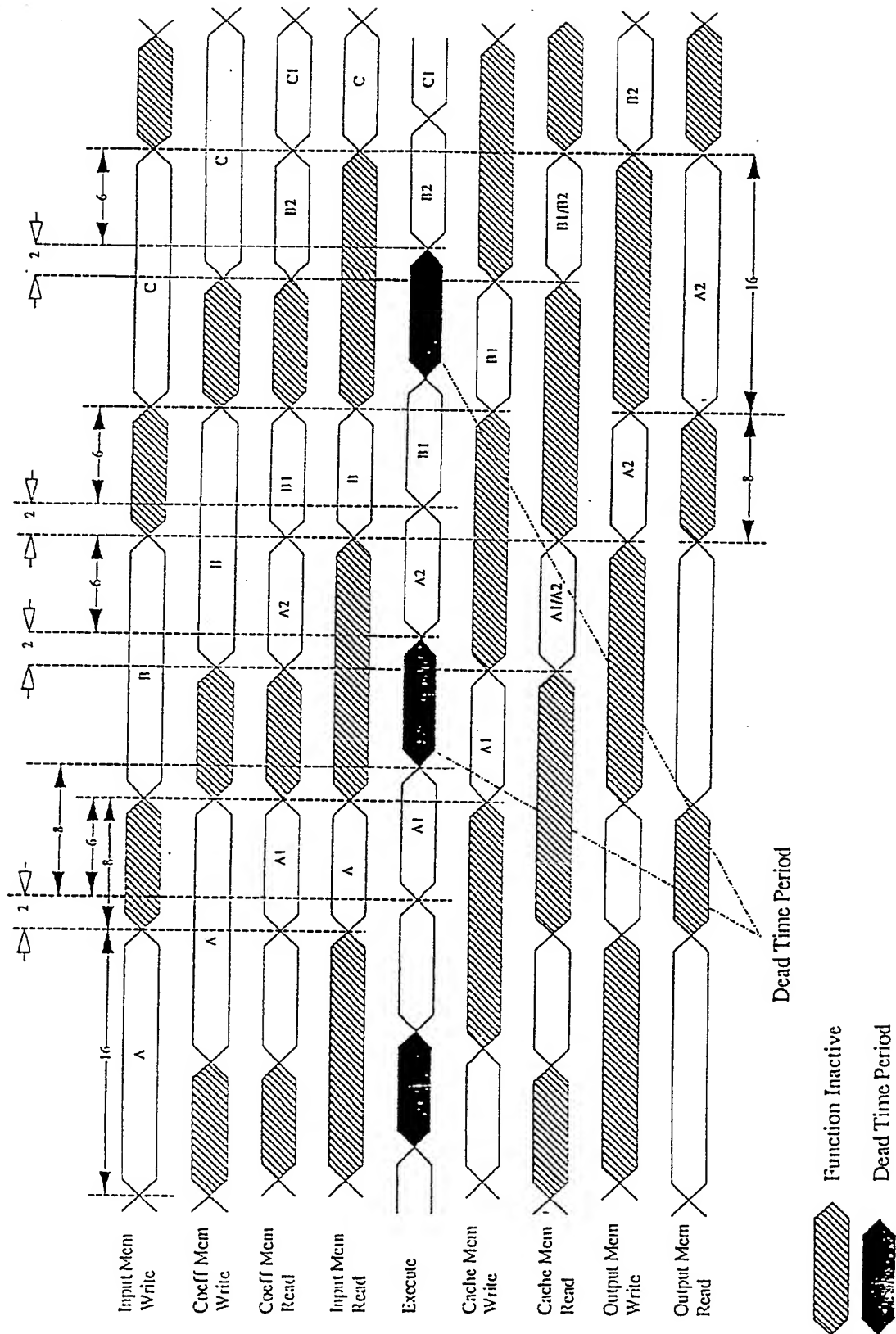D0=i; D1=D0+le1; D2=D1+le1; D3=D2+le1;

D4=D0+le; D5=D4+le1; D6=D5+le1 D7=D6+le1;

D8=D4+le; D9=D8+le1; DA=D9+le1; DB=DA+le1;

DC=D8+le; DD=DC+le1; DE=DD+le1; DF=DE+le1;

        }

    }

}

Twiddle Factor

$$w1= e^{\,j\frac{P*PI}{2*4^{m-1}}}$$

$$w4= w1^4$$

$$w7=w4*e^{\,j*PI*4^{m-2}}$$

$$w10=w7*e^{\,j*PI*4^{m-2}}$$

$$w13=w10*e^{\,j*PI*4^{m-2}}$$

$w2=w1^2$	$w3=w1^3$
$w5=w4^2$	$w6=w4^3$
$w8=w7^2$	$w9=w7^3$
$w11=w10^2$	$w12=w10^3$
$w14=w13^2$	$w15=w13^3$

P is equivalent to j in the upper data sequence

Figure 8 - N point FFT by QR16
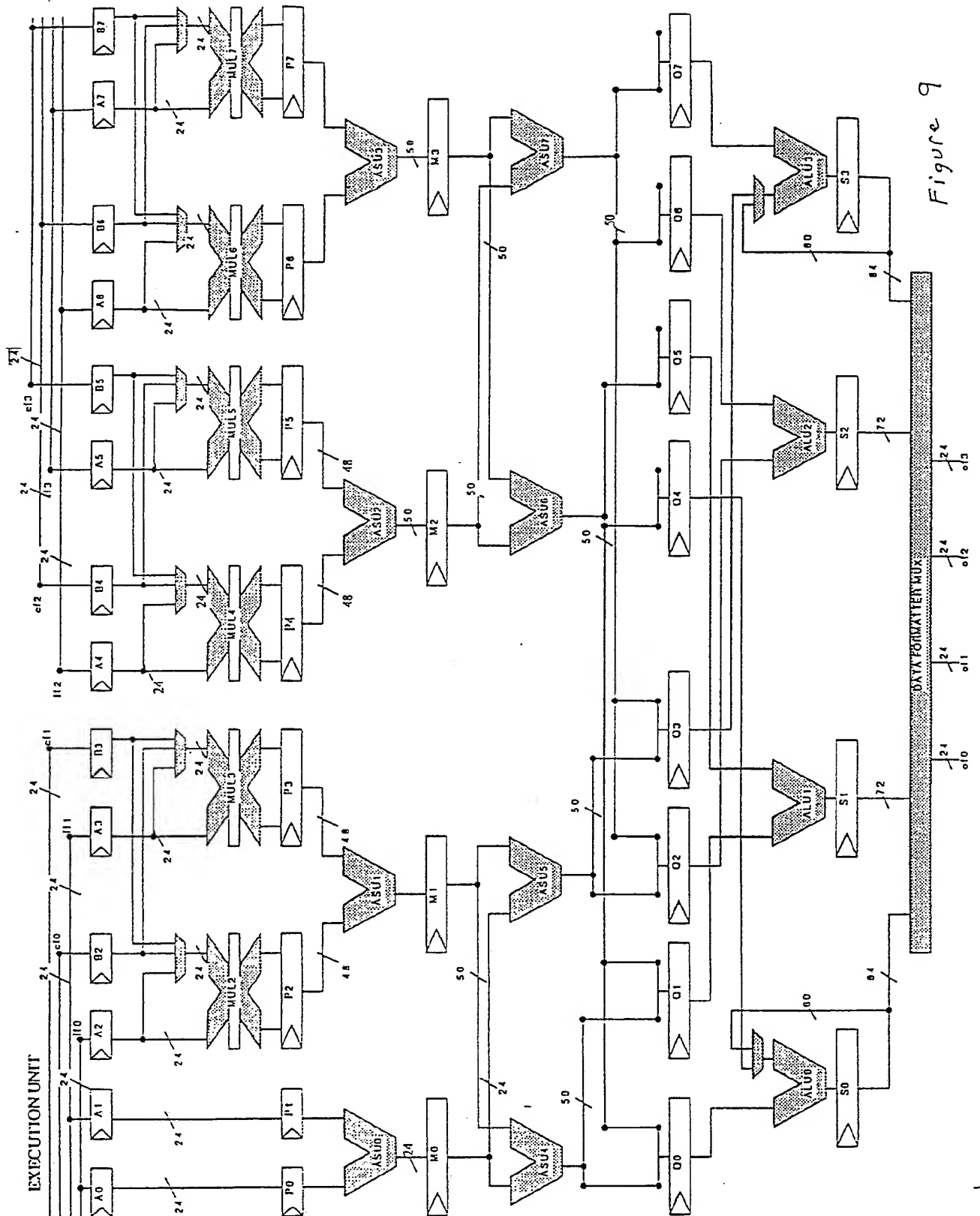
Figure 9

12/ 16

Figure 10 - System Environment

Notes: (1) ADD : Row address of memory
(2) A,B : Left bank of memory - For complex data, real data is written in bank A and imaginary data in bank B
(2) C,D : Right bank of memory - For complex data, real data is written in bank C and imaginary data in bank D

(a) Sequence 1

ADD.	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
A,B	0	2	4	6	8	a	c	e	10	12	14	16	18	1a	1c	1e
C,D	1	3	5	7	9	b	d	f	11	13	15	17	19	1b	1d	1f

(b) Sequence 2

ADD.	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
A,B																f
C,D	0	2	1	3	4	6	5	7	8	a	9	b	c	e	d	f

Figure 11 - Input Memory Write Sequences

Notes: (1) ADD : Row address of memory
     (2) A,B : Left bank of memory - For complex data, real data is written in bank A and imaginary data in bank B
     (2) C,D : Right bank of memory - For complex data, real data is written in bank C and imaginary data in bank D

(a) Sequence 1

ADD.	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	18	19
A,B	0	2	4	6	8	a	c	e	10	12	14	16	18	1a	1c	1e				
C,D	1	3	5	7	9	b	d	f	11	13	15	17	19	1b	1d	1f				

(b) Sequence 2

ADD.	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	18	19
A,B		1													f					
C,D	0	2	1	3	4	6	5	7	8	a	9	b	c	e	d	f				

(c) Sequence 3

ADD.	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f	10	11	18	19
A,B	0	5	8	b	e				15	18	1b					1e	0	2	10	12
C,D	4	6	7	9	a	c	d	f	14	16	17	19	1a	1c	1d	1f	1	3	11	13

Figure 12 - Coefficient Memory Write Sequences
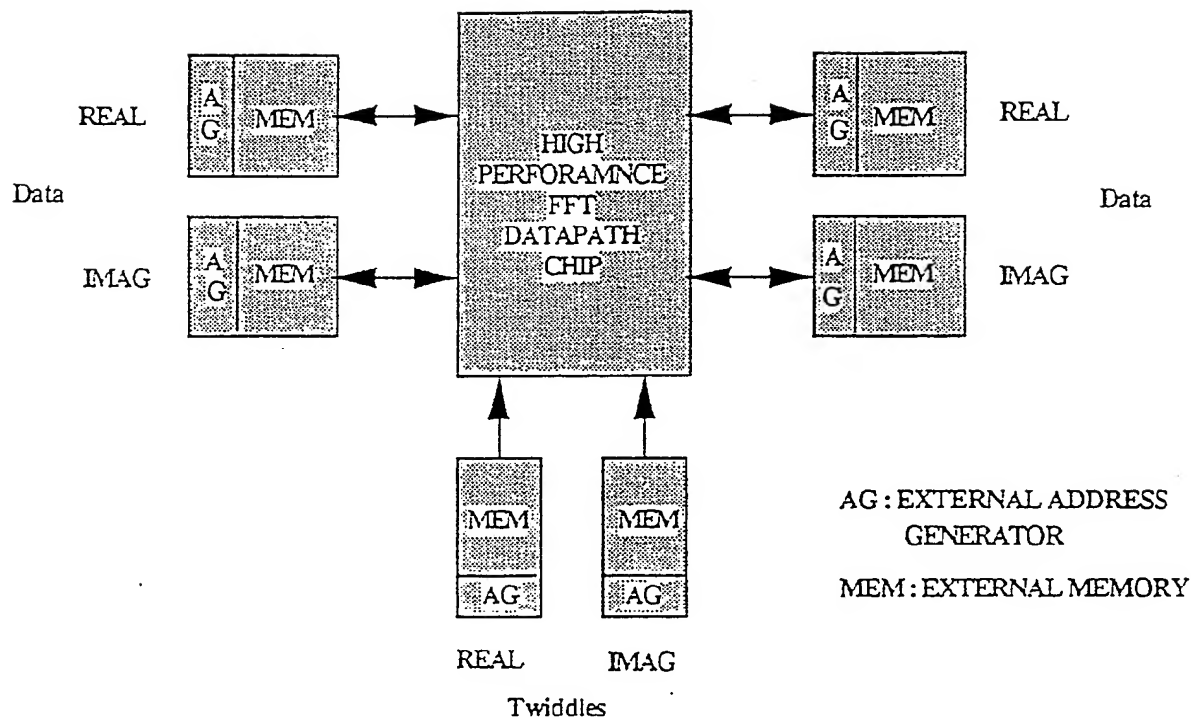
Notes: (1) ADD : Row address of memory
(2) A,B : Left bank of memory - For complex data, real data is written in bank A and imaginary data in bank B
(2) C,D : Right bank of memory - For complex data, real data is written in bank C and imaginary data in bank D

(a) Write Sequence

ADD.	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
A,B	0	8	5	d	2	a	7	1	10	18	15	1d	12	1a	17	1f
C,D	4	c	1	9	6	e	3	b	14	1c	11	19	16	1e	13	1d

(b) Read Sequence

ADD.	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
A,B	0	2	5	7	8	a	d	f	10	12	15	17	18	1a	1d	1f
C,D	1	3	4	6	9	b	c	e	11	13	14	16	19	1b	1c	1e

Figure 13 - Cache Memory Write/Read Sequences

Notes: (1) ADD : Row address of memory
(2) A,B : Left bank of memory - For complex data, real data is written in bank A and imaginary data in bank B
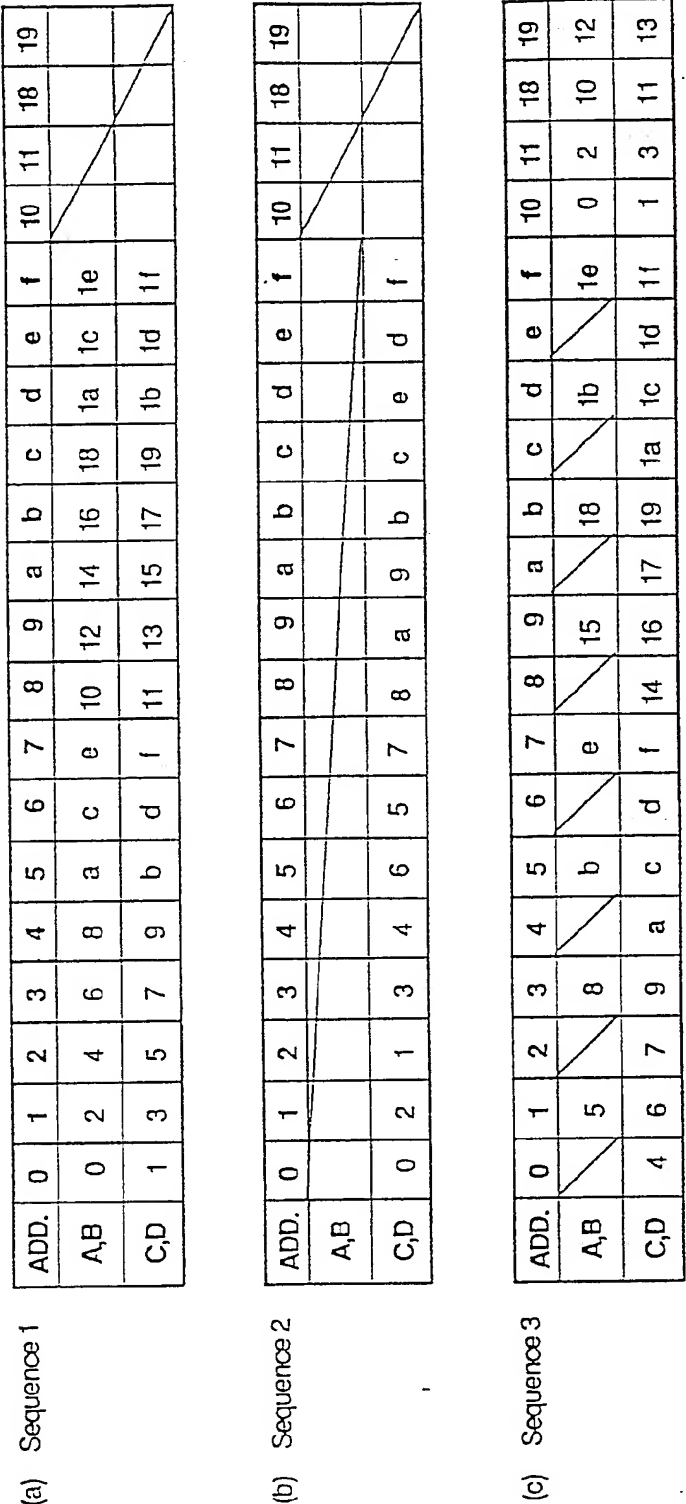(2) C,D : Right bank of memory - For complex data, real data is written in bank C and imaginary data in bank D

(a) Sequence 1

ADD.	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
A,B	0	1	4	5	8	9	c	d	10	11	14	15	18	19	1c	1d
C,D	2	3	6	7	a	b	e	f	12	13	16	17	1a	1b	1e	1f

(b) Sequence 2

ADD.	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
A,B	0	2	4	6	8	a	c	e	10	12	14	16	18	1a	1c	1e
C,D	1	3	5	7	9	b	d	f	11	13	15	17	19	1b	1d	1f

(c) Sequence 3

ADD.	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
A,B	0	8	1	9	2	a	3	b	10	18	11	19	12	1a	13	1b
C,D	4	c	5	d	6	e	7	f	14	1c	15	1d	16	1e	17	1f

Figure 14 - Output Memory Read Sequences

I. CLASSIFICATION OF SUBJECT MATTER   (if several classification symbols apply, indicate all)[6]

According to International Patent Classification (IPC) or to both National Classification and IPC

Int.Cl. 5 G06F15/332

II. FIELDS SEARCHED

Minimum Documentation Searched[7]

Classification System	Classification Symbols
Int.Cl. 5	G06F

Documentation Searched other than Minimum Documentation to the Extent that such Documents are Included in the Fields Searched[8]

III. DOCUMENTS CONSIDERED TO BE RELEVANT[9]

Category[o]	Citation of Document,[11] with indication, where appropriate, of the relevant passages[12]	Relevant to Claim No.[13]
X	IEEE TRANSACTIONS ON CIRCUITS & SYSTEMS , IEEE COMPUTER SOCIETY PRESS, NEW YORK, US vol. 35, no. 11, November 1988, pages 1434 - 1438; P.TORTOLI ET AL: 'A HIGH-SPEED FFT UNIT BASED ON A LOW COST DIGITAL SIGNAL PROCESSOR' see page 1435, left column, line 54 - line 59 see page 1435, right column, line 39 - line 47 see page 1436, right column, line 26 - page 1437, left column, line 33 see figures 1,2 --- -/--	1-10

[o] Special categories of cited documents :[10]

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

IV. CERTIFICATION

Date of the Actual Completion of the International Search	Date of Mailing of this International Search Report
24 JUNE 1992	.1 3. 07. 92

International Searching Authority	Signature of Authorized Officer
EUROPEAN PATENT OFFICE	BARBA M.

III. DOCUMENTS CONSIDERED TO BE RELEVANT	(CONTINUED FROM THE SECOND SHEET)	Relevant to Claim No.
Category°	Citation of Document, with indication, where appropriate, of the relevant passages	
A	EP,A,0 334 626 (DU PONT PIXEL SYSTEMS LIMITED, GB) 27 September 1989 see column 2, line 28 - line 40 see column 6, line 40 - line 44 see column 7, line 2 - line 57 see column 9, line 50 - column 10, line 31 see column 13, line 1 - line 35 see column 14, line 38 - column 15, line 10 see column 43, line 60 - column 44, line 15 ---	1-10
A	PROCEEDINGS OF 1988 IEEE INT. SYMP. ON CIRCUITS AND SYSTEMS, IEEE COMPUTERS SOCIETY PRESS, NEW YORK, US vol. 1/3, 7 June 1988, HELSINKI, FINLAND pages 73 - 76; N.A.RAMAKRISHNA ET AL: 'A TESTABLE CMOS SIGNAL PROCESSOR FOR FFT' see page 74, right column, line 1 - line 31 see page 75, left column, line 7 - line 15 see page 75, left column, line 40 - right column, line 3 ---	1-10
A	IEEE TRANSACTIONS ON COMPUTERS , IEEE COMPUTERS SOCIETY PRESS, NEW YORK, US vol. 36, no. 5, May 1987, pages 581 - 591; A.NORTON ET AL: 'PARALLELIZATION AND PERFORMANCE ANALYSIS OF THE COOLEY-TUKEY FFT ALGORITHM FOR SHARED MEMORY ARCHITECTURES' see page 582, left column, line 41 - right column, line 34 see page 583, right column, line 1 - line 15 see page 585, left column, line 15 - line 40 ---	1-10
A	ELECTRONIC DESIGN vol. 35, no. 18, 6 August 1987, HASBROUCK,NJ,US pages 101 - 106; K.LAMB: 'CMOS DESIGN BUILDING BLOCKS SHRINK AND SPEED UP FFT SYSTEMS' see page 104, left column, line 10 - line 16 see page 106, left column, line 2 - line 11 ---	1-10

This annex lists the patent family members relating to the patent documents cited in the above-mentioned international search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information. 24/06/92

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
EP-A-0334626	27-09-89	GB-A-	2217058	18-10-89
		JP-A-	2168351	28-06-90
		US-A-	5042000	20-08-91

EPO FORM P0479